Universal TCP: A Review of TCP Congestion Control Algorithms and TCP variants

ZafarIqbal Khan
Department of Computer Science & Engineering
Glocal University
Saharanpur, India

Dr Mohammad MazharAfzal
Department of Computer Science & Engineering
Glocal University
Saharanpur, India

***Abstract*—Transmission Control Protocol or TCP as it is widely known, forms a de-facto choice for transmission of packets from node to node. TCP provides connection oriented, reliable and end to end delivery links. Its implementation is not only and identical everywhere rather, it has certain variations. The Article presented here is a introduction and comparisons of those variants. How TCP performs in different scenarios is dependent of its implemented variant, which is used. The Study presented here is just an effort to identify feasibility of various variations of TCP into various conditions. Criterion being, congestion window, congestion avoidance, retransmission, fast recovery, acknowledgments, selective acknowledgments, and congestion control.**

Keywords-Mobile ad hoc networks, MANET, TCP, Transmission Control Protocol, congestion control, congestion window, selective acknowledgments

I. INTRODUCTION *(HEADING 1)*

TCP is a connection-oriented, end to end reliable protocol which has been designed to suit the layered hierarchy of protocols supporting multi-network applications [1]. TCP has been major carrier protocol since the inception of Internet [2]. with advent of technology and communication media used of TCP has been constant increase reaching to approximately 90% todays modern wired and wireless connected world [3]. Although communication technologies have come across significant development TCPs basic structure has not been change from its initial specification as in RFC 793, but still several enhancements were proposed by RFC 7414 [4]. As TCP working does not depend upon networks layer, during course of time several performance enhancements were proposed and subsequently incorporated into TCP leading to formation of TCP variants. Although suitability of these variants depends upon underlying usage scenarios and network infrastructures nevertheless, reliable flow control, congestion control, data transfer, connection manager, TCP segments and multiplexing are some issues keeping enhancement of TCP active.

Original TCP as defined in RFC 763 is well optimized for transporting datagrams into classical wired networks but caused some performance glitches in MANETs, as unreliable or broken links between node give retransmission and acknowledgement overheads to TCP. In wired networks packet loss was mainly due to congestion, whereas in wireless networks such as MANETs packet loss may occur due to a number of reasons. When packet loss is detected, tcp invokes congestion control routines which dramatically reduces packet window size. Proposed paper is a study of how different TCP variants have adopted to ad hoc, diverse and heterogeneous environments.

II. CONGESTION CONTROL

Procedural Architecture of TCP for Congestion Control
TCP has certain procedures defined for establishing host-to-host connection and securing datagrams delivery from sender to intended receiver node viz. Slow start, fast retransmission, fast recovery, congestion avoidance, congestion control and selective acknowledgement which are described briefly below.

A. Slow Start

It is a procedure which is incorporated by sender to control transmission rate also known as flow control on sender based. This is one of the basic routines available in all TCP variants implemented for sake of utilizing maximum bandwidth without causing congestion over communication channel. A sender starts communication at slow rate but increases it exponentially to achieve the maximum data transfer rate of communication channel between the sender and receiver [5]. This subroutine introduces a new state variable viz. Congestion window or cwnd and advertised window or rwnd. The congestion window (cwnd) is a limit imposed at sender end which checks on the amount of data any sender can sent through the network prior to reception of an acknowledgment (ACK), whereas the receiver's advertised window (rwnd) is a restriction on the amount of outstanding data at receiving end. The minimum of cwnd and rwnd controls rate of data transmission through the communication channel [3]. When a new connection is established with a node from some other network, the congestion window or cwnd is set to 1 segment, by default which is 512 [7]. When sent data is acknowledged by receiver, cwind is simply appended by 1, which causes exponential growth in cwnd size consequently reaching to the bandwidth limit of the communication channel. This causes some loss of acknowledgement packets raising conditions for congestion to TCP.

B. Congestion Avoidance

TCP starts communication with procedure of Slow Start. During Slow Start, cwnd is increased exponentially which sometimes results in congestion due to which many packets are dropped. Packet loss is due to two main reasons in any media, due to the damaged media and due to congestion in the media and full buffers in between sender and receiver respectively [5]. Therefore, to control the rate of transmission of segments procedure of Congestion Avoidance is used. This procedure further introduces three state variables viz. SMSS (Sender Maximum Segment Size), RMSS (Receiver Maximum Segment Size) and ssthresh (Slow Start Threshold). As defined in RFC2581 IW, the starting value of cwnd may take following size

$$IW = min(4*SMSS, max(2*SMSS, 4380\ bytes))$$

starting value of ssthresh may be arbitrary but most of the implementation choose size of advertised window .
Slow start algorithm is used when *cwnd<ssthresh*, while congestion avoidance is used when *cwnd>ssthresh*

During slow start, for each packet acknowledged TCP increments cwnd by at most SMSS bytes. Slow start ends when cwnd exceeds or equates ssthresh or when congestion is sensed.
During congestion avoidance, cwnd is augmented by 1 full-sized segment per round-trip time (RTT). Congestion avoidance continues until congestion is detected. cwnd values during congestion avoidance is as given

$$cwnd \mathrel{+}= SMSS*SMSS/cwnd$$

A combination of Slow Start and congestion avoidance [7] (two different approaches) is used to do again data transfer which has lost [11].

C. Fast Retransmission

Fast Retransmission is a mechanism adopted by TCP to procure lost packets from sender or to correct the out of order received packets and ensure the integrity of message being transmitted. It is designed to improve the recovery time in case of packet loss. The underlying principle is that, once a data segment is lost all the subsequent acknowledgments will carry the same next packet header or the sequence number of next packet, receiver is expecting to be sent by sender. These ACKs having same sequence number for next packet are called DUPACKs [6]. When sender receives 3 DUPACKs subsequently (4 in all, 1 usual + 3 duplicates), it retransmits the lost packet. As a optimizing measure, receiver sends DUPACKs immediately as soon as it receives an out of order packet.

D. Selective Acknowledgment (SACK)

Fast recovery as it is fine, but suffers a catastrophe when successive packet losses during a transmission reduce cwnd to min i.e. 2SMSS and communication starts creeping with frequent packet losses. As a corrective measure to overcome this problem TCP uses SACKs or Selective Acknowledgement which provides freedom to receiver to acknowledge only a few discontinuous data segments which were received successfully. A SACK can have, number of SACK Blocks containing starting and ending sequence numbers of data segments received successfully by receiver [8].

III. VARIANTS OF TCP

Based upon implementation of above procedures and algorithms, TCP has some major variations

A. TCP Tahoe

The specialty of TCP Tahoe is that it has incorporated Fast Retransmit Algorithm with slow start. When tahoe senses three duplicate ACKs which were not even piggybacked then it assumes that congestion has taken place and starts Fast Retransmit procedure, it further sets following changes

$$ssthresh=cwnd/2$$
$$cwnd=1\ SMSS$$

And starts slow start procedure [5]. Tahoe is the basis of all congestion control mechanisms available in TCP. Main drawback of Tahoe is that it consumes a complete round-trip to know that packet loss has occurred and congestion is taking place. During this all, communication channel becomes empty.

B. TCP Reno

TCP Reno is further refinement of TCP Tahoe, as in TCP Tahoe, Reno also relies upon duplicate ACKs from sender which are not piggybacked on data packets and also does not change advertised window size, then it assumes that congestion is taking place, it starts fast retransmit procedure, and makes following changes to state variables

$$cwnd=cwnd/2$$
$$ssthresh=cwnd$$

Unlike Tahoe which makes new congestion window to 1 SMSS, Reno makes it just half of previous size and starts another mechanism known as Fast Recovery [8].

C. TCP Vegas

TCP Vegas is further enhancement of congestion avoidance which rather than focusing upon packet loss, considers packet delays. By considering packet delays TCP Vegas determines optimal rate of packet transmission.
Rerouting of packets changes RTT of rerouted packets and hence cause serious overheads for TCP Vegas, as TCP Vegas excessively relies upon Round Trip Time for sent packets, it's accurate calculation is extremely important, under calculated values of RTT will lead to sub optimal use of available channel bandwidth, whereas an overrated value will make futile overall effort [9].

D. TCP New Reno

The TCP Reno was designed to supplant the original TCP Tahoe model and subsequently had a considerably more effective congestion control procedure [10]. This also includes a fast recovery state to avoid the congestion window decline for the initial state after three duplicate ACKs, which takes too long to recover the original window size. The fast recovery state reduced the congestion window by just half and starts a congestion avoidance mechanism [11]. The SACK or Selective acknowledgement option includes, recognition to non-adjoining blocks of data, which identifies each block with a header number, containing first and last communicated segments [10].

The New Reno brought some new features to fast recovery algorithm viz. partial acknowledgements which are received while the fast recovery procedure is carried out. Partial Acknowledgements are those ACKs which do not ensure receiving of all data packets which has been sent from sender which initiates retransmission of first unacknowledged segment. This partial acknowledge keeps congestion window unaltered which otherwise have been resulted into TCP invoking fast recovery mode [11].

E. TCP Hybla

TCP Hybla was evolved from the necessities of long range radio links or satellite communication channels which have large RTT. Previous versions of TCP were dependent upon RTT values to invoke congestion control and avoidance algorithms. TCP Hyblaconsists some analytical procedures to eliminate those excessive dependencies [12].

F. TCP BIC

TCP BIC stands for (Binary Increase Congestion control), which derives from its unique congestion control mechanism optimized for special type of networks called Long Fat Networks characterized by their high speed and latency. BIC incorporates binary search algorithms to find the optimum size of congestion window (cwnd) where it can remain unaltered for relatively long period of time [13].

G. TCP CUBIC

TCP CUBIC is a refinement of TCP BIC, with cwnd is a cubic function of time since the last congestion incident, we set point of inflection just prior to the event. Use of cubic function gives TCP CUBIC a special curve when cwnd plotted with time. Initially cwnd grows rapidly to the size just before the last congestion event, this is a concave part of the plot, and then at this point of the time it remains for relatively long time before starting to probe for more available bandwidth from the channel, this steady part gives stability to the communication. Next phase is a convex growth region where CUBIC probes for more available bandwidth [14].

Congestion window cwnd for TCP CUBIC is given by

$$W_{cubic} = C(t-K)^3+W_{max}$$

Here, C is scaling factor, t is time elapsed from last window reduction, W_{max} is the congestion window size prior to last time window was reduced.

H. TCP Westwood

TCP Westwood is the modification of TCP Reno at sender side in the sense that it estimates the whole bandwidth from sender to receiver. These modifications make TCP Westwood suit better for networks with large bandwidth delays, transmission pipes with potential packet losses, leaky pipes and channels with dynamic loads. Westwood mines acknowledgement streams to set optimal values for congestion control parameters viz. cwnd and ssthresh [15].

I. TCP Proportional Rate Reduction

TCP Proportional Rate Reduction (PRR) is the refinement introduced by Google corp. to enhance the accuracy of data transmitted during recovery process. This algorithm is designed to make window size cwnd as closer to ssthresh as possible [16].

J. TCP BBR

TCP BBR (Bottleneck Bandwidth and Round trip propagation time) is a novel model based approached designed by Google corp. to estimate optimal window size for heterogeneous wide networks [17]. As todays network consist bandwidths from 2G to 4G and network interface controllers are capable of tackling traffic from Mbps to Gbps, key idea is packet loss should not be only criterion to identify congestions. This model based adaptive approach for congestion control gives better understanding of network traffic and congestion [18]. Each time a cumulative acknowledgment is received, a rate sample is obtained from it, and showing, what amount data was delivered over the time elapsed, and acknowledgment were received [19].

CONCLUSION

Over the time Internet and communication networks have evolved enormously and TCP remained most widely used Transport Layer Protocol. TCP has also evolved with time adapting itself to changing behavior of users and networks but a large proportion of world population is still facing scarcity of network bandwidth in developing countries, with average network speed of 114 kbps and 2.5G network [20]. Hence designing of universally suitable TCP which performs optimal in all environments is still a goal to achieve.

REFERENCES

[1] *Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981. https://tools.ietf.org/html/rfc793* **Error! Hyperlink reference not valid.**

[2] *https://tools.ietf.org/html/rfc7414#page-9*

[3] *http://www.redbooks.ibm.com/pubs/pdfs/redbooks/gg243376.pdf*

[4] *Jacobson, V., "Congestion Avoidance and Control", Computer Communication Review, vol. 18, no. 4, pp. 314-329, Aug. 1988. ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z* **Error! Hyperlink reference not valid.**

[5] *Stevens, W., "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", RFC 2001, January 1997.*

[6] *Floyd, S. and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm", RFC 2582 April 1999.*

[7] *Stevens, W., "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", RFC 2001, January 1997*

[8] *Fall, K. and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno and SACK TCP", Computer Communication Review, July 1996. ftp://ftp.ee.lbl.gov/papers/sacks.ps.Z***Error! Hyperlink reference not valid.**.

[9] *Kurose, James; Ross, Keith (2008). Computer Networking – A Top-Down Approach (4th ed.). Addison Wesley. ISBN 978-0-13-607967-5.*

[10] *M. Mathis, et al. "Request for comments 2883 - an extension to the selective acknowledgement (sack) option for tcp". Pittsburgh Supercomputing Center, July 2012. http://tools.ietf.org/search/rfc2883 .*

[11] *T. Henderson, et al. "Request for comments 6582 - the new reno modification to tcp's fast recovery algorithm". University of Oulu, April 2012. http://tools.ietf.org/pdf/rfc6582.pdf*

[12] *http://hybla.deis.unibo.it/*

[13] *BIC and CUBIC Networking Research Lab. https://research.csc.ncsu.edu/netsrv/?q=content/bic-and-cubic*

[14] *Sangtae Ha, Injong Rhee and LisongXu, CUBIC: A New TCP-Friendly High-Speed TCP Variant, ACM SIGOPS Operating System Review, Volume 42, Issue 5, July 2008, Page(s):64-74, 2008. http://www4.ncsu.edu/~rhee/export/bitcp/cubic-paper.pdf*

[15] *http://c3lab.poliba.it/index.php?title=Westwood*

[16] *Proportional Rate Reduction for TCP https://tools.ietf.org/html/rfc6937*

[17] *BBR: Congestion Based Congestion Control – ACM Queue, http://queue.acm.org/detail.cfm?id=3022184*

[18] *BBR: Congestion Control – IETF, https://www.ietf.org/proceedings/97/slides/slides-97-iccrg-bbr-congestion-control-02.pdf*

[19] *Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, SoheilHassasYeganeh, Van Jacobson, "BBR: Congestion – Based Congestion Control", ACM Queue, vol. 14, September-October (2016), pp. 20 – 53, 2016*

[20] *TCP BBR congestion control comes to GCP – your Internet just got faster https://cloudplatform.googleblog.com/2017/07/TCP-BBR-congestion-control-comes-to-GCP-your-Internet-just-got-faster.html*